Durante as aulas usamos `bind()` para ligar (*bind*) o valor de `this` a um determinado contexto. Existem outros dois métodos para manipular o contexto de `this`:

`call()`

O método `call()` executa a função passando valores e parâmetros específicos para serem usados como contexto do `this`. Ou seja, é possível atribuir um `this` diferente do contexto atual ao executar a função.

Um exemplo de uso de `call()` para especificar o contexto de `this`:

```
function exibeInfos() {
 console.log(this.nome, this.email)
}

const user = {
 nome: 'Mariana',
 email: 'm@m.com'
}

exibeInfos.call(user)
```

Fazendo com que a função seja executada em determinado contexto, mesmo após ser instanciada:

```
function User(nome, email) {
 this.nome = nome
 this.email = email

 this.exibeInfos = function(){
   console.log(this.nome, this.email)
 }
}

const newUser = new User('mariana', 'm@m.com')
```

```
const outroUser = {
 nome: 'Rodrigo',
 email: 'r@r.com'
}

newUser.exibeInfos() //mariana m@m.com
newUser.exibeInfos.call(outroUser) //Rodrigo r@r.com
```

Também é possível passar parâmetros para `call()`, como no exemplo a seguir.

Temos uma função que monta uma determinada mensagem a partir dos parâmetros `nome` e `email`. Se quiséssemos vincular os dados da mensagem a um objeto com dados de usuários, podemos usar `call()` passando como primeiro parâmetro o contexto a ser considerado como `this` (no caso, objeto `user`) e a partir do segundo parâmetro definimos quais os argumentos.

```
function exibeMensagem(nome, email) {
 console.log(`usuário: ${nome}, email ${email}`)
}
 const user = {
 nome: 'Mariana',
 email: 'm@m.com',
 executaFuncao: function(fn) {
   fn.call(user, this.nome, this.email)
 }
}

user.executaFuncao(exibeMensagem) //usuário: Mariana, email m@m.com
```

Nesse caso, a função que será executada também está sendo passada como parâmetro de `executaFuncao` e usamos `call()` para "chamar" a função com um contexto (`this`) específico e também argumentos específicos.

`apply()`

O método `apply()` funciona de forma semelhante ao `call()`, porém recebe a lista de argumentos em um array:

```
function exibeMensagem(nome, email) {
 console.log(`usuário: ${nome}, email ${email}`)
}
 const user = {
 nome: 'Mariana',
 email: 'm@m.com',
 executaFuncao: function(fn) {
   fn.apply(user, [this.nome, this.email])
 }
}
 user.executaFuncao(exibeMensagem) //usuário: Mariana, email m@m.com
```

Usando arrays, é possível passar os argumentos via variável ou até mesmo usando a propriedade `arguments` que existe internamente em todo objeto.